Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Lab 3: Digital Modulation & Transmission - II

1. Carrier Synchronisation and Symbol Timing Recovery in BPSK

The simulation in Section Lab 2: Section 3 uses a baseband representation of a BPSK communication system. That is, the process of modulating the message information onto a high frequency sinusoidal carrier, as is typically required in wireless communication, was omitted from our simulation. As we discussed in the lecture, the frequency up-conversion process (translating the baseband message to a high-frequency carrier using a mixer) done in the transmitter can be exactly undone by the frequency downconversion process (translating the message information imparted on the high-frequency carrier down to baseband using the same mixer) in the receiver. Therefore, our simulations are valid for wireless systems under the assumption that the frequency translation processes are ideal.

For coherent communication systems, the key challenge is to ensure synchronisation between the transmitter and receiver. In this part of the laboratory you will observe and investigate coherent digital modulation techniques using real-world wireless signals.

In particular, you will observe:

- Carrier Synchronisation: the process of ensuring that the receive carrier is synchronised, in both frequency and phase, to the transmit carrier.
- Symbol Timing Recovery: the process of ensuring that the receiver precisely knows the symbol timing so that it may determine when to sample the recovered message for making symbol decisions.

Both of these processes are essential for coherent communication. Symbol timing recovery, but not carrier synchronization, is required in non-coherent communication systems.

Since we are using SDRs, these processes are implemented through adaptive algorithms in software. Traditional radios use hardware circuitry to achieve the functionality. Our focus will be on observing and understanding (and seeing!) the importance of these processes within a coherent digital communication system, rather than the algorithms or circuits through which they are implemented. If interested in learning about these algorithms, you are encouraged to check detailed materials in the references of the course textbook.

Download the BPSK Receiver Simulink model from blackboard. The model is shown in Figure 2. The blocks in the model have been pre-configured for this laboratory exercise.

IMPORTANT: For the BPSK receiver to work properly, the carrier frequencies of the transmitter and receiver must be relatively close to each other (ideally, no more than a few hundred Hz to perhaps 1kHz apart). You may have to compensate the center frequency of the PLUTO-SDR Receiver block in the model for accurate results. For our experiment, this can be accomplished either the Frequency Correction (ppm) setting, while with a calibration process, you can de-tune the center frequency according based on the observations during the calibration run (usually involving a known stable frequency source like a Pilot symbol in Digital Television broadcast [in the US]) . Doing so provides a coarse frequency correction. A fine frequency correction will be applied through an adaptive algorithm in the Simulink model. Outside of this, do not modify any other (advanced) configuration parameter of the PLUTO-SDR Receiver block.

It is given that:

- The BPSK symbols are being transmitted at 20000 bits per second.

- The receiver model performs coarse frequency correction using the Frequency Correction (ppm) setting of the PLUTO-SDR Receiver block.
  - Sampling rate: 300e3
  - Output data type: single
  - Samples per frame: 55200

- The receive gain is controlled via a **Slider Gain** block, making it easy to adjust the receive gain while the model is running.

- The **AGC** block implements an automatic gain control system. This block adaptively adjusts its gain to give a constant signal level at its output. Such a block is common in systems in which the receiver input signal level changes due to variation in the communication channel.

- The **Carrier Synchronizer** block adjusts for carrier frequency and phase offsets in the incoming

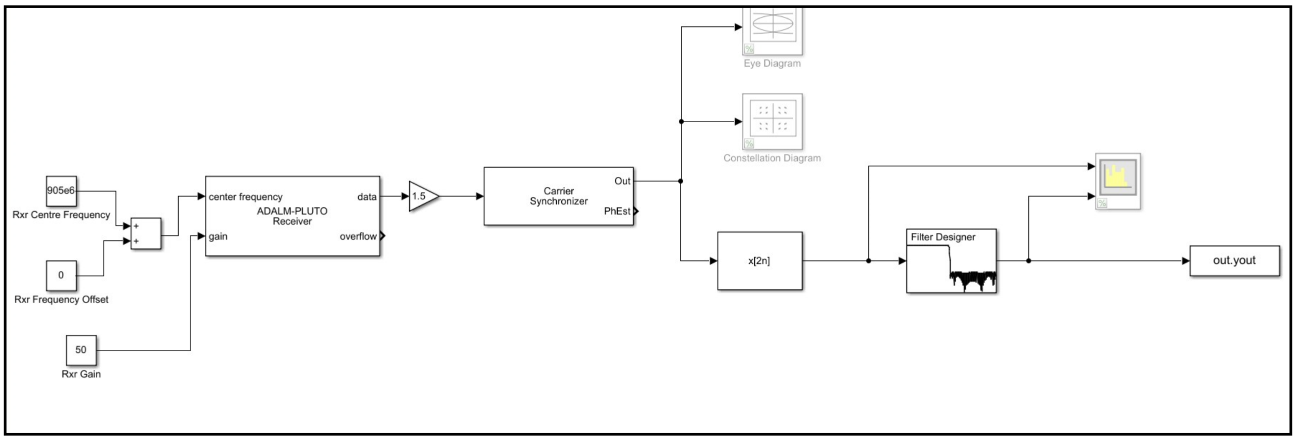

Fig 2: Simulink model for a DBPSK receiver using the PLUTO device.

  signal. This block will compensate for offsets that remain after coarse calibration of the carrier frequencies. It is be configured to observe the data with 15 samples per symbol, with damping factor 0.707 and loop bandwidth of 0.01. Modulation type is BPSK with phase left to auto.

- The **Raised Cosine Receive Filter** block filters the input signal using a raised cosine pulse shape filter. The block is configured to match the pulse shape used in your instructor's transmitter, and therefore mimics a matched filter. The block performs a decimation of 3.

- The **Symbol Synchronizer** block adjusts for symbol clock drift in the incoming signal, ensuring that incoming pulses are sampled (for purposes of deciding which symbol was sent) at the "opening" of the eye. The block is configured to output exactly one sample per symbol interval. The block hence also performs decimation by 5.

- The **BPSK Demodulator Baseband** block demodulates its input signal and outputs the recovered bit sequence, which can be viewed on the Display block.

Question 1.1: Observe the Eye Diagram plots at the input and output of the Carrier Synchronizer block. Using a screenshot of the plots, explain any observations and/or interpretations.

Question 1.2: Observe the Constellation Diagram plots at the input and output of the Carrier Synchronizer block. Using a screenshot of the plots, explain any observations and/or interpretations.

Question 1.3: Observe the Constellation Diagram plots at the input and output of the Symbol Synchronizer block. Using a screenshot of the plots, explain any observations and/or interpretations.

Question 1.4: Observe the effects of reduced signal-to-noise ratio (SNR) at the input to the receiver by

- decreasing the Gain of the PLUTO-SDR Receiver
- placing an object near/over the receive antenna

Examine the effects of reduced SNR on Constellation Diagram and Eye Diagram plots. Take screen capture of your observations and provide any inferences you may have.

---

2. Differential BPSK

In section 4 of previous lab, you designed and tested a coherent receiver for BPSK communication. In most cases, phase-shift keying requires coherent detection due to the inherent nature of the modulation; however, a technique called differential phase-shift keying allows for phase-shift keyed symbols to be detected without a fully coherent receiver. The key here is that the D-PSK receiver uses the immediately prior symbol as a phase reference to demodulate the current symbol.

In this section, you will use D-BPSK to receive and decode a message signal that has an ASCII text as the message content. Appendix B covers the theory on D-BPSK modulation

2.1. D-BPSK Simulation

NOTE: This section does not require you to use the PLUTO device or the transmitter in the lab, hence, you can come back to this section at the end of part 1.2.

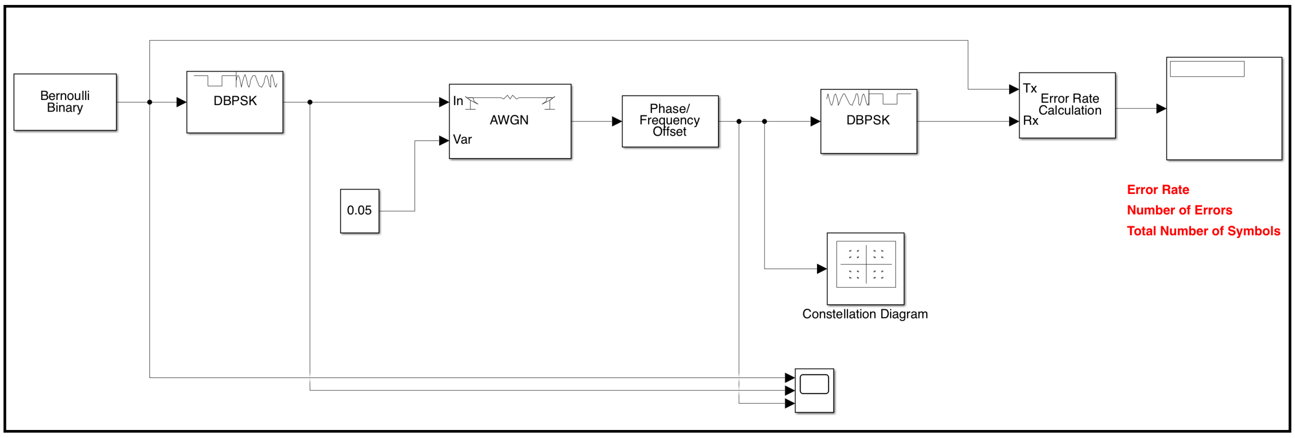

Fig 1: Simulink model for a D-BPSK communication

Download the `dbpsk-simuation.slx` simulation model from BB. The model is shown in Fig 1. The key blocks in the design are explained below along with any observations you could make from the simulation

- The model simulates D-BPSK using symbols. It does not include pulse shaping in the transmitter or matched filtering in the receiver (although you can add these elements). Set the bit rate to 8000 bits per second.

- The **D-BPSK Modulator Baseband** block applies differential encoding followed by BPSK modulation to the binary message generated by the Bernoulli Binary Generator block. It uses the differential encoding scheme shown in Table 1 (Appendix B). Carefully examine the Time Scope plot to see the message bits and the differentially encoded symbols.

- The **Phase/Frequency Offset** block is used to apply frequency offsets Δf and phase offsets ϕ to the received symbols. Based on the constellation diagram, can you explain what these offsets are?

- At what frequency offset do you begin to see large numbers of errors? Why is this the case?

- The **Constant** block controls the channel noise power. Experiment with the toggle to see the impact of channel noise.

2.2. D-BPSK Demodulator

In this section, you will design a D-BPSK demodulator to decode a text message that was received by a Pluto Receiver. Each of you have an assigned .mat file corresponding to your board number which you should download from Blackboard. Your task is to decode the DBPSK symbols and recover the text by writing a MATLAB script.

- Import the message_rx mat file into your workspace using the load command along with the preamble. When imported, you will see a structure named `yout` in the MATLAB workspace. The actual DBPSK symbols are located within the structure at `yout.message_rx.signals.values`

To recover the ASCII text, follow the hints below:

- Write a MATLAB script in the workspace to recover the ACSII text. This can be done outside of the lab since baseband data is made available to you.

- Recall that received bits are differentially encoded, and hence you will need to decode them first. Use functions like `angle` to determine the angle associated with each symbol. Functions like `wrapToPi` and `wrapTo2Pi` might come in handy while working with angles.

- A message *preamble* within the message needs to be decoded to identify the start of the data frame, so that you can properly group the bits into ASCII characters. Recall that the preamble is known (loaded via the preamble mat file), so think of what function could be useful in this case.
- The message frame is constructed as below

  [guard preamble preamble data guard]
  - Guard band is established by 100 samples of 0's
  - Preamble is a 128 sample Barker Sequence
  - Data is the message that was transmitted. It is unique for each person.
  - Each sample is 1 bit (binary encoding)

- Distance between successive preambles should give you length of the message (including the preambles and guard bands).

- Multiple messages may be present in the same

```matlab
%% Convert text into a vector of bits
a = dec2bin('Test Message\n',8);     % get binary representation; gives character array
numChars = size(a,1);                % get number of characters

a = a - '0';                         % converts char array into matrix of numbers
a = reshape(a.',8*numChars,1);       % reshape matrix into a vector; 8 bits per
    character

stem(a)                              % plot vector of bits

%% Recover text from vector of bits
b = reshape(a, 8, numChars).';       % reshape vector into matrix; 8 bits per row
b = num2str(b);                      % convert to char array
b = bin2dec(b);                      % convert to decimal
fprintf(char(b));                    % print characters
```

Fig 3: MATLAB script example for converting between ASCII characters and vectors of binary data

- The below snippet (Fig 3) shows a scheme for converting between ASCII text and a vector of bits.

Question 2.1: Differential encoding is quite useful to overcome the phase noise issues that you encountered with standard BPSK. However, what could be a potential issue with building them using the Pluto devices for our lab?

Question 2.2: Describe in a few lines your method for decoding the differentially encoded symbols. Include the salient parts of your MATLAB code you wrote to perform the decoding.

Question 2.3: What is the text message?

Submission

Submit the following for your observation as a <u>short report</u> via blackboard:

- Screenshot/captures and notes from all sections
- Screen capture of the signals, constellations accordingly
- Observations from BPSK receiver section (1.1)
- Observations from simulating the DBPSK baseband system (section 2.1)
- Answers to questions 1.1 through 2.3
- Include code snippets for decoding the binary data in section 2.2

Appendix A

We can consider a DBPSK modulator that forms its communication signal using the following two operations:

A. The binary message data is differentially encoded. Let {b$_k$} denote the binary message data and {d$_k$} denote the differentially encoded message. One example of differential encoding operates as follows:
   a. if the current symbol b$_k$ is a 0, then leave the symbol d$_k$ the same as the previous symbol d$_{k-1}$
   b. if the current symbol b$_k$ is a 1, then change the symbol d$_k$ with respect to the previous symbol d$_{k-1}$

This example of differential encoding is illustrated in the table below where the initial encoded symbol (at k = –1) is arbitrarily chosen to be 1.

Table 1: Differential encoding example.

k	-1	0	1	2	3	4	5	6	7
$\{b_k\}$		1	0	0	1	0	1	1	0
$\{d_k\}$	1	0	0	0	1	1	0	1	1

B. The differentially encoded message is phase-shift keyed. For the case of BPSK with rectangular pulse shapes, the complex envelope is

$$z(t) = A \times m(t)$$

where m(t) has values ±1 over each symbol interval based on {d$_k$}. The passband BPSK waveform is

$$\begin{aligned}
s(t) &= \Re\{z(t)\exp(j2\pi f_c t)\} \\
&= \Re\{Am(t)\exp(j2\pi f_c t)\} \\
&= Am(t)\cos(2\pi f_c t) \\
&= \begin{cases} A\cos(2\pi f_c t), & m(t) = 1 \\ -A\cos(2\pi f_c t), & m(t) = -1 \end{cases} \\
&= \begin{cases} A\cos(2\pi f_c t), & m(t) = 1 \\ A\cos(2\pi f_c t + \pi), & m(t) = -1 \end{cases}
\end{aligned}$$

The key result here is that the message information is encoded in the phase difference between consecutive BPSK symbols. The D-BPSK receiver then examines the relative phase between consecutive symbols to recover the message.

We now consider our standard model for an SDR receiver. Because the receive carrier is not synchronised to the transmit carrier, we model the receiver as having frequency offset Δf and phase offset ϕ. Therefore, the receiver output is

$$\begin{aligned}
\hat{z}(t) &= \mathrm{LPF}\{s(t)e^{-j(2\pi(f_c-\Delta f)t+\phi)}\} \\
&= \mathrm{LPF}\{Am(t)\cos(2\pi f_c t)e^{-j(2\pi(f_c-\Delta f)t+\phi)}\} \\
&\vdots \\
&= \frac{1}{2}Am(t)e^{j(2\pi\Delta f t-\phi)}
\end{aligned}$$

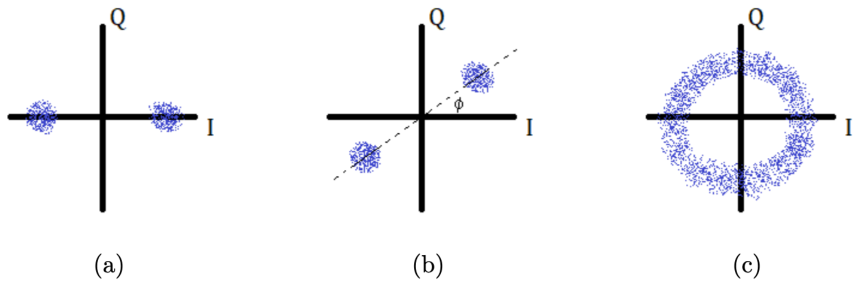

Fig 6: DBPSK constellations for (a) coherent detection, (b) carrier phase error, and (c) carrier frequency error.

Consider the very fortunate, but highly improbable, case in which $\Delta f = 0$ and $\phi = 0$ which gives

$$\hat{z(t)} = \frac{1}{2}A \times m(t)$$

This case corresponds to coherent detection. Notice that, as expected, the differentially encoded symbols are perfectly recovered! This case is illustrated in the (noisy) constellation diagram shown in Fig 6 (a).

Next consider another highly improbable case where $\Delta f = 0$ and $\phi \neq 0$. That is, the receive carrier has only a phase error. The receiver output is

$$\hat{z(t)} = \frac{1}{2}A \times m(t)e^{j\phi}$$

which means that the symbols have been rotated in the I/Q space by angle ϕ. However, the message bits are recoverable since the symbols retain the proper relative phase (π radians). This case is illustrated in the constellation diagram shown in Figure (b).

Finally, consider the practical and likely case where $\Delta f \neq 0$ and $\phi \neq 0$. That is, the receive carrier has both a frequency and phase error. The receiver output is

$$\hat{z(t)} = \frac{1}{2}A \times m(t)e^{j(2\pi\Delta ft - \phi)}$$

which means that the symbols are undergoing rotation in the I/Q space at the rate $2\pi\Delta f$ radians per second. So long as Δf is small compared to the symbol rate, the phase change between consecutive symbols will remain close to either 0 or π, thus preserving the differential encoding. This case is illustrated in the constellation diagram shown in Figure (c).